19Z610 - MACHINE LEARNING LABORATORY

STROKE PREDICTION

PSG COLLEGE OF TECHNOLOGY

TEAM - 7

Team Members

20Z204 - Adarsh G
20Z215 - Elanthamil R
20Z220 - Jeevan Krishna K V
20Z267 - Nirmal M
21Z431 - Ajay Deepak P M

PROBLEM STATEMENT

Stroke is the second largest cause of mortality worldwide and remains an enormous health burden for individuals. Hypertension, heart illness, diabetes and dysregulation of glucose metabolism, atrial fibrillation, and lifestyle variables are some of the controllable risk factors of stroke. The objective of our project is to successfully predict a person's likelihood of suffering a stroke based on potentially modifiable risk variables by applying machine learning methods to big data sets. This may be done by analyzing medical records using machine learning models to find patterns that are related to the risk of stroke.

DATASET DESCRIPTION

The dataset is provided to the model which predicts whether or not a patient will experience a stroke based on the input parameters. Each row in the data contains relevant information about a different patient including gender, age, various diseases etc. There are eleven columns that contain attributes related to each individual's data and the last column contains the patient's likelihood of suffering a stroke. The attributes are explained as follows[1].

The dataset we use consists of 5111 rows and 12 columns.

- Id : Unique Identifier.
- Gender : "Male", "Female" or "Other"
- Age : Age of the patient.
- Hypertension : 1 if the patient has hypertension or 0 if not.
- Heart_disease : 1 if the patient has a heart disease or 0 if not.
- Ever_married : "No" or "Yes".
- Work_type : "Never_worked", "Children", "Govt_job", "Private", "Self_emp".
- Residence_type : "Rural" or "Urban".
- Avg_glucose_level : Average glucose level in blood.
- Bmi : body mass index.
- Smoking_status : "Formerly smoked", "Never smoked", "Smokes" or "Unknown".
- Stroke : 1 if the patient will have a stroke or 0 if not.

id	gender	age	hypertension	heart_disease	ever_married	work_type	Residence_type	avg_glucose_level	bmi	smoking_status	stroke
9046	Male	67	0	1	Yes	Private	Urban	228.69	36.6	formerly smoked	1
51676	Female	61	0	0	Yes	Self-employ	Rural	202.21	N/A	never smoked	1
31112	Male	80	0	1	Yes	Private	Rural	105.92	32.5	never smoked	1
60182	Female	49	0	0	Yes	Private	Urban	171.23	34.4	smokes	1
1665	Female	79	1	0	Yes	Self-employ	Rural	174.12	24	never smoked	1
56669	Male	81	0	0	Yes	Private	Urban	186.21	29	formerly smoked	1
53882	Male	74	1	1	Yes	Private	Rural	70.09	27.4	never smoked	1
10434	Female	69	0	0	No	Private	Urban	94.39	22.8	never smoked	1
27419	Female	59	0	0	Yes	Private	Rural	76.15	N/A	Unknown	1
60491	Female	78	0	0	Yes	Private	Urban	58.57	24.2	Unknown	1
12109	Female	81	1	0	Yes	Private	Rural	80.43	29.7	never smoked	1
12095	Female	61	0	1	Yes	Govt_job	Rural	120.46	36.8	smokes	1
12175	Female	54	0	0	Yes	Private	Urban	104.51	27.3	smokes	1
8213	Male	78	0	1	Yes	Private	Urban	219.84	N/A	Unknown	1
5317	Female	79	0	1	Yes	Private	Urban	214.09	28.2	never smoked	1
58202	Female	50	1	0	Yes	Self-employ	Rural	167.41	30.9	never smoked	1
56112	Male	64	0	1	Yes	Private	Urban	191.61	37.5	smokes	1
34120	Male	75	1	0	Yes	Private	Urban	221.29	25.8	smokes	1

Fig.1: Snippet of our dataset

METHODOLOGY/MODELS USED

The models used for the project are:

- Decision Trees: A decision tree is a type of supervised learning algorithm used in machine learning and data mining. It is a tree-like model that represents a set of decisions and their possible consequences or outcomes.
- Random Forest: A decision tree is a type of supervised learning algorithm used in machine learning and data mining. It is a tree-like model that represents a set of decisions and their possible consequences or outcomes.
- Naive Bayes: Naive Bayes classification is a probabilistic machine learning algorithm used for classification tasks. It is based on Bayes' theorem, which states that the probability of a hypothesis(such as a class label) is proportional to the probability of the evidence(such as the observed features) given that hypothesis.
- Logistic Regression: Logistic regression is a statistical machine learning algorithm used for classification tasks. It is a type of regression analysis where the dependent variable is binary(i.e. it can take only two values such as 0 or 1).
- Neural Network: A neural network is a type of machine learning algorithm inspired by the structure and function of the human brain. It consists of interconnected nodes, also known as neurons, organized into layers. Each neuron receives input from one or more other neurons, performs a computation, and passes the output to other neurons in the network.
- KNN: The algorithm works by finding the K nearest training samples(neighbors) in the feature space to the new input data point and assigning a label or value based on the majority class or average value of the K neighbors.

TOOLS USED

1. **Language** **:** Python
2. **Coding Platform** **:** VS Code / Google Colab / Spyder
3. **Libraries** **:**

- Numpy - A general-purpose package for handling arrays. In addition to tools for working with these arrays, it offers a high-performance multidimensional array object. It has a wide range of features, such as a potent N-dimensional array object, sophisticated functions, tools for using linear algebra, the Fourier transform, and random number capabilities[6].
- Pandas - Pandas is an open-source library designed primarily for working quickly and logically with relational or labeled data. It offers a range of data structures and procedures for working with time series and numerical data. The NumPy library serves as the foundation for this library[6].
- Scikit-learn - An open-source Python toolkit which uses a uniform interface to implement a variety of machine learning, pre-processing, cross-validation, and

visualization methods. It has a number of clustering, regression, and classification techniques, including as support vector machines, random forests, gradient boosting, and k-means, among others[6].

- Matplotlib - A multi-platform data visualization package to deal with the larger SciPy stack and is based on NumPy arrays. One of visualization's biggest advantages is that it gives us visual access to vast volumes of data in forms that are simple to understand. There are various plots in Matplotlib, including line graphs, bar graphs, scatter plots, and histograms[6].
- Seaborn - A graphical statistical plot visualization library for Python. For constructing a variety of statistical graphs in Python, Seaborn offers numerous color schemes and default styles. Basically, charts are used to show how different variables relate to one another. Relational plots, category plots, distribution plots, regression plots, matrix plots, and multiplot rasters are the several types of plots that Seaborn categorizes[6].

CHALLENGES FACED

- Data availability and quality: One of the biggest challenges faced in stroke prediction is acquiring a sufficient and diverse dataset. Additionally, poor data quality and incomplete or incorrect data in the dataset can have a detrimental effect on accuracy of prediction.
- Unbalanced data: The dataset used to predict stroke can be unbalanced, meaning that one class (positive stroke cases, for example) may have a disproportionately high number of examples compared to its counterpart (negative stroke cases). As a result, models may become skewed and accuracy of prediction may decline.
- Model interpretability: Some machine learning models are difficult to interpret, which can be a challenge in a medical field where decisions have significant impact on patients' health.
- Model selection and tuning: There are many different types of machine learning algorithms, each with its own strengths and weaknesses. Choosing the right algorithm for a given problem and optimizing its parameters can be a difficult task, especially for complex datasets.
- Ethical and legal considerations: Stroke prediction models could have great privacy implications, and it's far more important to make sure that the records are safely maintained and moral concerns are taken into account. Additionally, there are legal requirements for obtaining consent for using personal data in such models.

CONTRIBUTION OF TEAM MEMBERS

20Z204 - Adarsh G	Data Preprocessing, Model Development
20Z215 - Elanthamil R	Data Collection, Testing
20Z220 - Jeevan Krishna K V	Model Development, Testing
20Z267 - Nirmal M	Data Collection, Data Preprocessing
21Z431 - Ajay Deepak P M	Data Preprocessing, Model Development

ANNEXURE I: CODE

```
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.metrics import mean_squared_error
#The above code segment import libraries for preprocessing

stroke_data=pd.read_csv('/healthcare-dataset-stroke-data.csv')
stroke_data.head()
/displays the dataset
stroke_data.dtypes
/displays the data types of features in the dataset
categorical_features = stroke_data.select_dtypes(include=['object']).columns
numerical_features = stroke_data.select_dtypes(include=['int64', 'float64']).columns
print(categorical_features)
print(numerical_features)
stroke_data .shape
stroke_data .info()
stroke_data.describe()
duplicate_Values=stroke_data.duplicated()
print(duplicate_Values.sum())
stroke_data[duplicate_Values]
```

```
stroke_data.isnull().sum()
stroke_data = stroke_data.drop('id',axis=1)
Stroke_data
```

#The above code segment handles differentiation of categorical and numerical variables and duplicates in dataset

```
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(stroke_data.iloc[:,8].array.reshape(-1, 1))
stroke_data.iloc[:,8]= imputer.transform(stroke_data.iloc[:,8].array.reshape(-1, 1))
```

#The above code segment handles missing data in BMI using mean strategy

```
labelencodergen = LabelEncoder()
stroke_data.gender = labelencodergen.fit_transform(stroke_data.gender)
labelencodermar = LabelEncoder()
stroke_data.ever_married = labelencodermar.fit_transform(stroke_data.ever_married)
labelencoderwork= LabelEncoder()
stroke_data.work_type = labelencoderwork.fit_transform(stroke_data.work_type)
labelencoderres = LabelEncoder()
stroke_data.iloc[:, 6]= labelencoderres.fit_transform(stroke_data.iloc[:, 6])
labelencodersmoke = LabelEncoder()
stroke_data.iloc[:, 9] = labelencodersmoke.fit_transform(stroke_data.iloc[:, 9])
stroke_data
```

#The above segment handles categorical data by label encoding then using different encoders

```
import matplotlib.pyplot as plt
import seaborn as sns
for col in ['age','avg_glucose_level','bmi']:
        sns.boxplot(x=stroke_data[col])
        plt.show()
```

#The above segment visualizes the outliers in the numerical columns of the dataset

```
def remove_outlier(df_in, col_name):
        q1 = df_in[col_name].quantile(0.25)
        q3 = df_in[col_name].quantile(0.75)
        iqr = q3-q1
        low  = q1 - 1.5*iqr
```

```
        high = q3 + 1.5*iqr
        df_out = df_in.loc[(df_in[col_name] > low) & (df_in[col_name] < high)]
return df_out
for i in ['age','avg_glucose_level','bmi']:
        stroke_data = remove_outlier(stroke_data,i)
        stroke_data.shape
for col in ['age','avg_glucose_level','bmi']:
        sns.boxplot(x=stroke_data[col])
        plt.show()
stroke_data.stroke.value_counts()
```

#The above code segment removes outliers in the numerical columns

```
corr = stroke_data.corr()
corr
plt.figure(figsize=(10, 10))
sns.heatmap(corr, cmap="RdYlGn", annot=True)
plt.show()
```

#The above code segment visualises the correlation matrix using a heatmap

```
X=stroke_data.drop(['stroke'],axis=1)
Y=stroke_data.iloc[:,-1].values
```

#The above code segment divides the dataset into dependent and independent variables

```
from sklearn.compose import ColumnTransformer
ct = ColumnTransformer(transformers=[('encoder', OneHotEncoder(), [0,2,3,4,5,6,9])],
remainder='passthrough')
X = np.array(ct.fit_transform(X))
```

#The above code segment converts categorical data into semantically acceptable form using one hot encode

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(X)
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state =9)
```

#The above code segment splits the dataset into test and train data

/Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.tree import DecisionTreeClassifier
clf = RandomForestClassifier(max_depth=50, random_state=0,n_estimators=50)
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred) - Accuracy : 0.95744
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
cm
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

/Logistic Regression

```
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression(random_state=0)
logreg.fit(X_train,y_train)
y_pred = logreg.predict(X_test)
accuracy_score(y_test,y_pred) - Accuracy : 0.96009
cm = confusion_matrix(y_test, y_pred)
cm
print(classification_report(y_test,y_pred))
```

#Since the above classifiers were biased because of the imbalances in the dataset oversampling was necessary to create a reliable model

```
from collections import Counter
from sklearn.datasets import make_classification
from imblearn.over_sampling import RandomOverSampler
print(Counter(Y))
oversample = RandomOverSampler(sampling_strategy='minority')
X_over, Y_over = oversample.fit_resample(X, Y)
print(Counter(Y_over))
X_trainover, X_testover, y_trainover, y_testover = train_test_split(X_over, Y_over, test_size =
.2, random_state = 42)
```

#The above code segment selects random samples from the minority and duplicates them to equal the majority

```
accuracy = []
preci=[]
recall = []
f1 = []
def algodetails(rep):
        accuracy.append(rep['accuracy'])
        preci.append(rep['macro avg']['precision'])
        recall.append(rep['macro avg']['recall'])
        f1.append(rep['macro avg']['f1-score'])
```

#The above code segment takes note of the classifiers properties in the forthcoming segment

```
/Random Forest Classifier
ranclfover= RandomForestClassifier(max_depth=50, random_state=0,n_estimators=50)
ranclfover.fit(X_trainover, y_trainover)
y_predover = ranclfover.predict(X_testover)
accuracy_score(y_testover,y_predover) - Accuracy : 0.99407
print(classification_report(y_testover, y_predover))
report = classification_report(y_testover, y_predover,output_dict=True)
algodetails(report)
cm = confusion_matrix(y_testover, y_predover)
cm
```

```
/Logistic Regression
from sklearn.linear_model import LogisticRegression
Logreg = LogisticRegression()
Logreg.fit(X_trainover, y_trainover)
y_predover = Logreg.predict(X_testover)
print(classification_report(y_testover, y_predover))
report = classification_report(y_testover, y_predover,output_dict=True)
algodetails(report)
cm = confusion_matrix(y_testover, y_predover)
cm
```

/Naive Bayes Classifier

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
gnb.fit(X_trainover, y_trainover)
y_predover = gnb.predict(X_testover)
print(classification_report(y_testover, y_predover))
report = classification_report(y_testover, y_predover,output_dict=True)
algodetails(report)
cm = confusion_matrix(y_testover, y_predover)
cm
```

/Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
dest = DecisionTreeClassifier(max_depth=50)
dest.fit(X_trainover, y_trainover)
y_predover = dest.predict(X_testover)
print(classification_report(y_testover, y_predover))
report = classification_report(y_testover, y_predover,output_dict=True)
algodetails(report)
cm = confusion_matrix(y_testover, y_predover)
```

/Neural Network

```
from sklearn.neural_network import MLPClassifier
nn = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 2), random_state=1)
nn.fit(X_trainover, y_trainover)
y_predover = nn.predict(X_testover)
print(classification_report(y_testover, y_predover))
report = classification_report(y_testover, y_predover,output_dict=True)
algodetails(report)
cm = confusion_matrix(y_testover, y_predover)
cm
```

/Support Vector Machine

```
from sklearn.svm import SVC
svma = SVC()
svma.fit(X_trainover, y_trainover)
```

```
y_predover = svma.predict(X_testover)
print(classification_report(y_testover, y_predover))
report = classification_report(y_testover, y_predover,output_dict=True)
algodetails(report)
cm = confusion_matrix(y_testover, y_predover)
cm
```

```
/KNN
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(3)
knn.fit(X_trainover, y_trainover)
y_predover = knn.predict(X_testover)
print(classification_report(y_testover, y_predover))
report = classification_report(y_testover, y_predover,output_dict=True)
algodetails(report)
cm = confusion_matrix(y_testover, y_predover)
cm
```

```
/Algorithm comparison
algo_liste=["Random Forest","Logistic Regression","Naive Bayes","Decision Tree","Neural
Networks","Support Vector Machine","KNN"]
score={"algorithms":algo_liste,"Accuracy":accuracy,"precision":preci,"recall":recall,"f1_score":f
1}
df = pd.DataFrame(score)
```

#The above code segment prints the properties of different classifiers

```
f, ax1 = plt.subplots(figsize =(15,15))
plt.title('Stroke Prediction Dataset',fontsize = 20,color='blue')
plt.figure(1)
sns.pointplot(x=df['algorithms'], y=df['Accuracy'],data=df,color='lime',label="Accuracy")
f,ax1 = plt.subplots(figsize =(15,15))
plt.figure(2)
sns.pointplot(x=df['algorithms'], y=df['precision'],data=df,color='red',label="precision")
f,ax1 = plt.subplots(figsize =(15,15))
plt.figure(3)
sns.pointplot(x=df['algorithms'], y=df['recall'],data=df,color='black',label="recall")
```

```
f,ax1 = plt.subplots(figsize =(15,15))
plt.figure(4)
sns.pointplot(x=df['algorithms'], y=df['f1_score'],data=df,color='blue',label="f1_score")
plt.xlabel('Algorithms',fontsize = 15,color='blue')
plt.xticks(rotation= 45)
```

#The above code segment visualizes the properties of all classifiers in a graph plot

```
plt.grid()
plt.legend()
plt.show()
tt = {"gender":['Male'],  "age":[67], "hypertension":[1],"heart_disease":[1]
,"ever_married":['Yes'],  "work_type":['Private'],  "Residence_type":['Urban'],
"avg_glucose_level":[228.60], "bmi":[36.6], "smoking_status":['smokes']}
tt = pd.DataFrame(tt)
tt.iloc[:, 0] = labelencodergen.transform(tt.iloc[:, 0])
tt.iloc[:, 4] = labelencodermar.transform(tt.iloc[:, 4])
tt.iloc[:, 5] = labelencoderwork.transform(tt.iloc[:, 5])
tt.iloc[:, 6]= labelencoderres.transform(tt.iloc[:, 6])
tt.iloc[:, 9] = labelencodersmoke.transform(tt.iloc[:, 9])
tt = np.array(ct.transform(tt))
tt
ranclfover.predict(tt)
```

#The above code segment is used as a means of inputting data into a classifier

ANNEXURE II: SNAPSHOTS OF THE OUTPUT

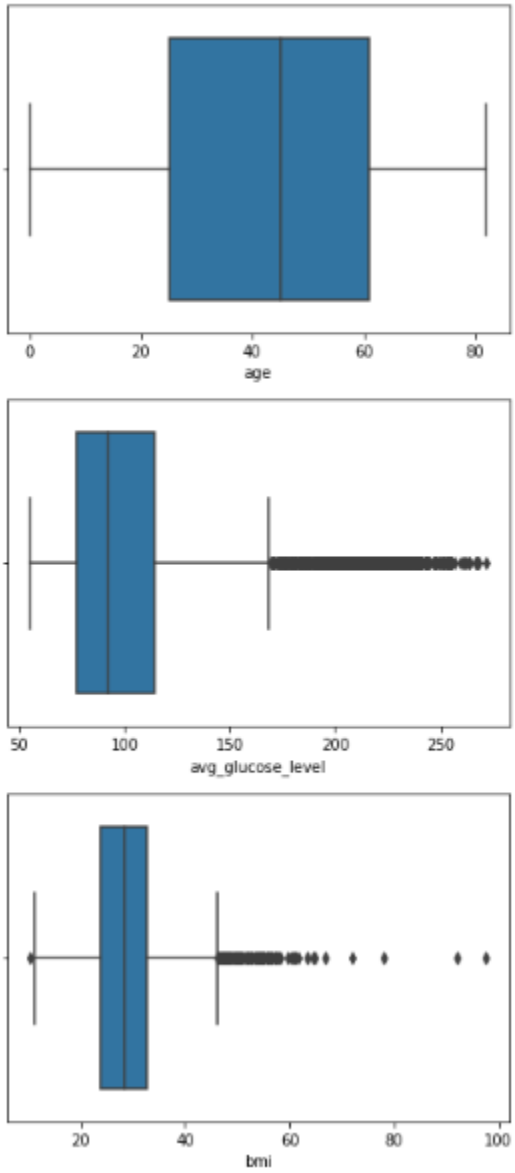

Fig.2: Box Plot of numerical columns showing many outliers

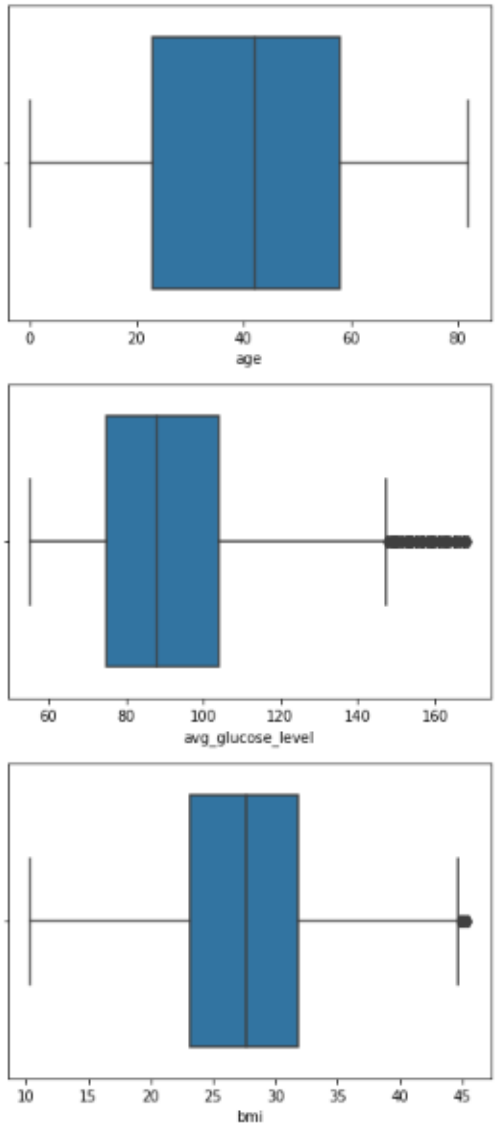

Fig.3: Box Plot of numerical columns showing few outliers

	algorithms	Accuracy	precision	recall	f1_score
0	Random Forest	0.994076	0.994186	0.994033	0.994075
1	Logistic Regression	0.777251	0.777948	0.777063	0.777023
2	Naive Bayes	0.626777	0.763810	0.624198	0.568574
3	Decision Tree	0.980450	0.981314	0.980310	0.980436
4	Neural Networks	0.815758	0.825268	0.815147	0.814192
5	Support Vector Machine	0.846564	0.851563	0.846139	0.845898
6	KNN	0.956754	0.960455	0.956444	0.956644

Fig.4: Comparison of different algorithms

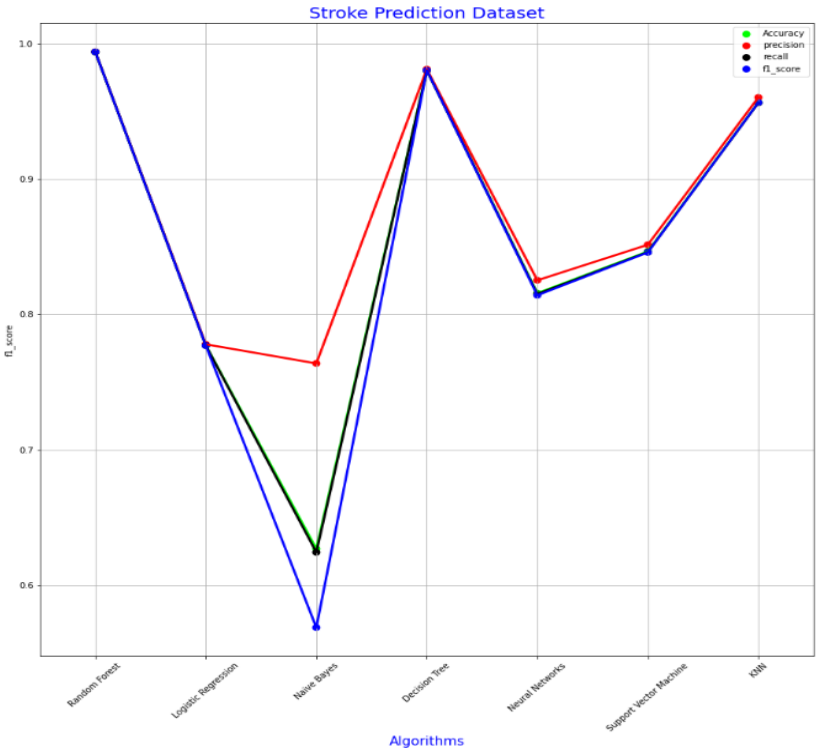

Fig.5: Plot of scores of different algorithms

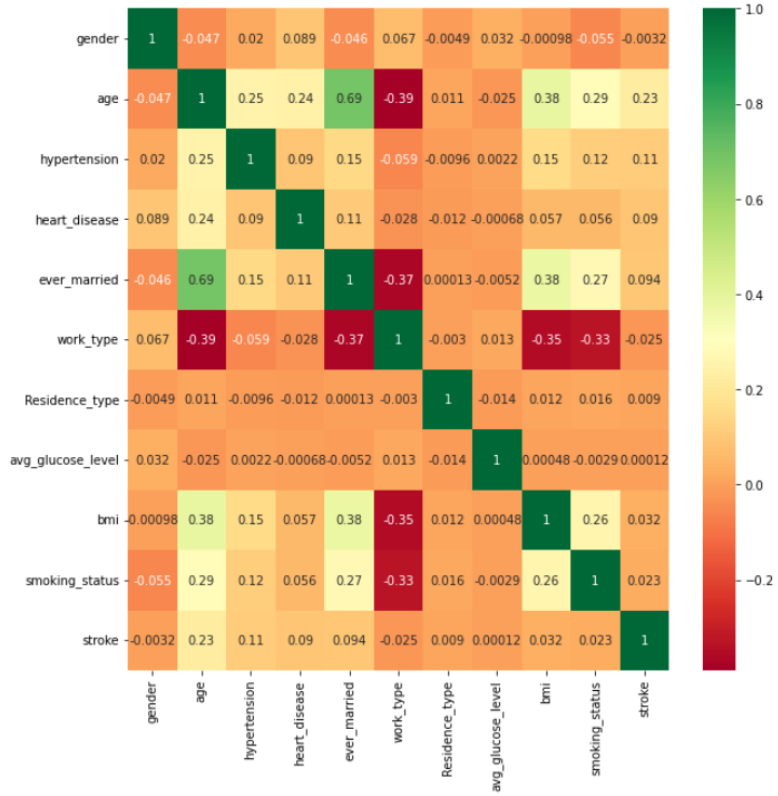

Fig.6: HeatMap of the correlation matrix of the entire dataset

REFERENCES

[1] Dataset named ‘Stroke Prediction Dataset’ from Kaggle: https://www.kaggle.com/fedesoriano/stroke-prediction-dataset

[2] Gangavarapu Sailasya and Gorli L Aruna Kumari, “Analyzing the Performance of Stroke Prediction using ML Classification Algorithms”, International Journal of Advanced Computer Science and Applications, Vol. 12, No. 6, 2021.

[3] Everything You Need to Know About Stroke: https://www.healthline.com/health/stroke

[4] S. Gupta and S. Raheja, "Stroke Prediction using Machine Learning Methods," 2022 12th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2022, pp. 553-558, doi: 10.1109/Confluence52989.2022.9734197.

[5] Stroke: Causes, Risk Factors, Treatment and Prevention: https://youtu.be/E_3LSi8QOKA

[6] Libraries in Python: https://www.geeksforgeeks.org/libraries-in-python/

[7] N. S. Adi, R. Farhany, R. Ghina and H. Napitupulu, "Stroke Risk Prediction Model Using Machine Learning," 2021 International Conference on Artificial Intelligence and Big Data Analytics, Bandung, Indonesia, 2021, pp. 56-60, doi: 10.1109/ICAIBDA53487.2021.9689740.

[8] Data Preprocessing: https://www.javatpoint.com/data-preprocessing-machine-learning

[9] Stroke: www.mayoclinic.org/diseases-conditions/stroke/symptoms-causes/syc-20350113

[10] Data Preprocessing: https://www.scalablepath.com/data-science/data-preprocessing-phase